Programmeren in AutoLISP

Les een

De Visual LISP Editor

Een korte handleiding

door

Joop F. Moelee

een gelovig volger van
“de Sacrale Kunst van Luiheid”
zijn Hoge Priester LISP en Acoliet Script

Copyright © 2004 by Joop F. Moelee

Permission to use, copy, modify, and distribute this document and the software it contains for any purpose and without fee is here by granted, provided that the above copyright notice appears in all copies and that both that copyright notice and the limited warranty and restricted rights notice below appear in all supporting documentation.

The writer of this document provides this document and the program code contained in this document “as is” and with all its faults.
The writer of this document specifically disclaims any implied warranty of merchantability or fitness for a particular use.
The writer of this document does not warrant that the operation of the code contained in this document will be uninterrupted or error free.

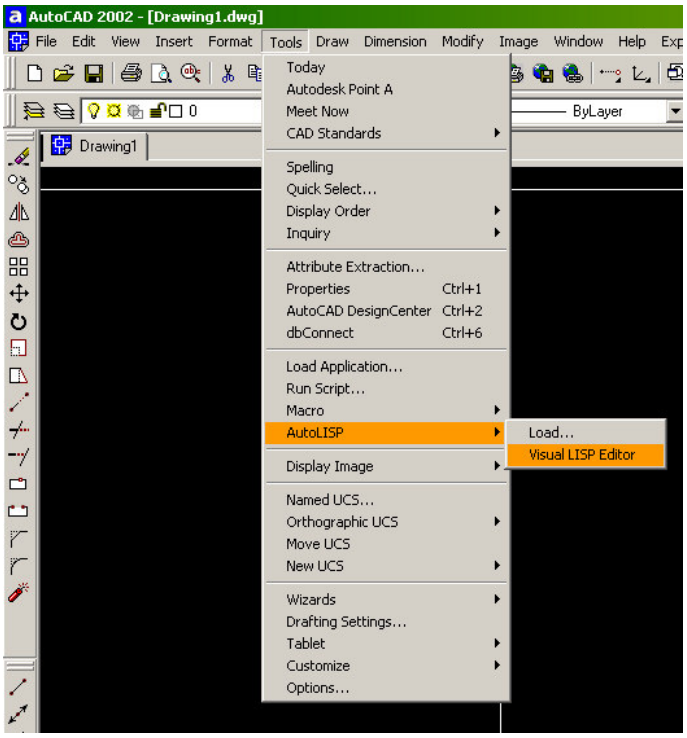
1) Inhoudsopgave

2) Verantwoording

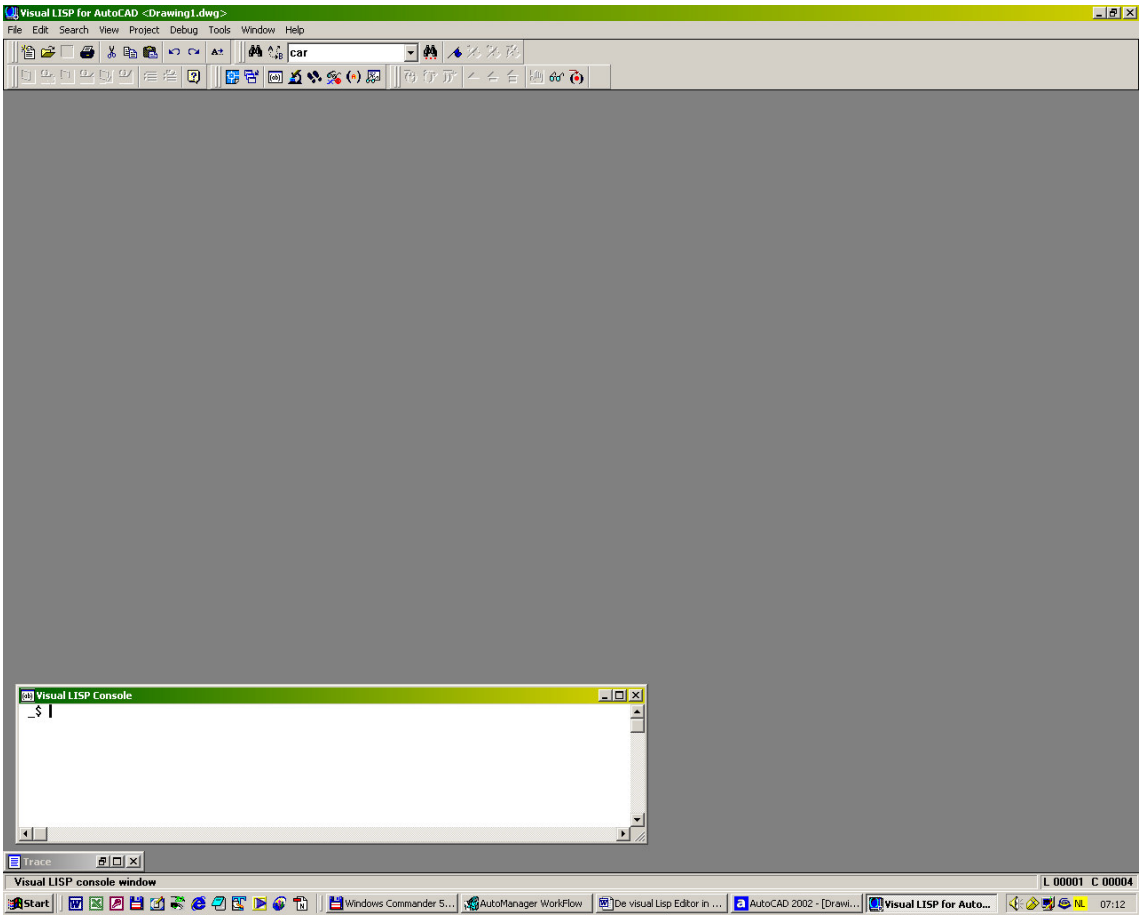
Dit document is gebaseerd op en/of bevat delen van de artikelen geschreven door Kenny Ramage eigenaar/ beheerder van de site http://www.afraLISP.com/
Kenny heeft de schrijver dezes toestemming gegeven tot gebruik van zijn teksten.
Hiervoor heel hartelijk bedankt, Kenny.
Op deze webpagina zijn vele interessante en leerzame artikelen te vinden over o.a. Auto LISP en Visual LISP. Ze hebben allen echter een nadeel: ze zijn geschreven in het Engels, en niet iedereen kan hier goed mee uit de voeten. Zeker de beginnend programmeur zal moeite hebben met de, soms ingewikkelde, technische Engelse teksten.
Van diverse kanten en over langere tijd heb ik het verzoek ontvangen een cursus te schrijven over het programmeren in LISP. En wel in het Nederlands.
Bij deze dan deel een.

3) Visual LISP Editor starten

We gaan het hebben over de Visual LISP Editor.
Het is nu niet de bedoeling de Visual LISP Editor in alle details te behandelen. De bedoeling is de belangrijkste functies te behandelen zodat we ermee kunnen werken. De rest wordt gedurende de diverse lessen in detail behandeld.
Genoeg gekletst, laten we beginnen. Start AutoCAD en open een nieuwe tekening.
Ga naar menu *TOOLS→AutoLISP→Visual LISP Editor*

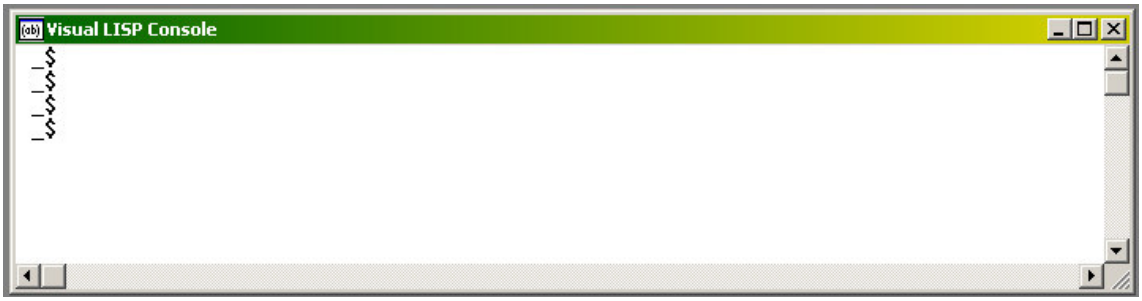


De *Visual LISP Editor* opent in een venster en ziet er ongeveer zo uit:



4) Console Window

We kijken eerst naar de *Console Window*:



De *Visual LISP Console* lijkt in sommige opzichten op de *AutoCAD Command Window*, maar heeft een paar extra's. Men typt tekst in de *Console Window* achter de *Console Prompt* die er zo uit ziet:

_$

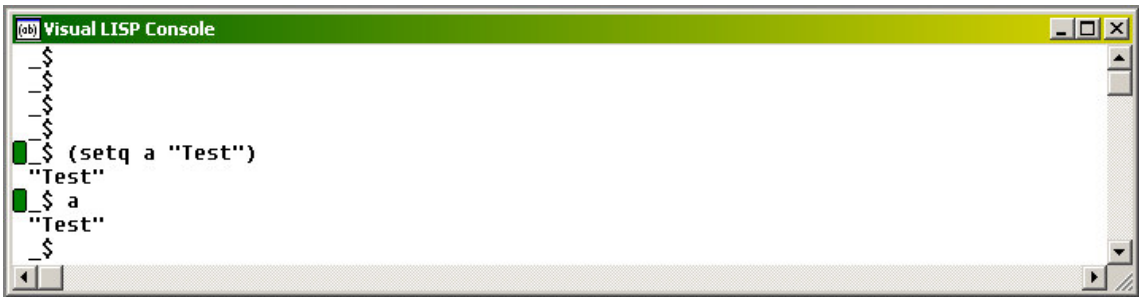
Type het volgende achter de *Console Prompt* en druk dan op de *Enter* toets:

_$ (setq a “Test”)

Type nu dit en druk weer op de *Enter* toets:

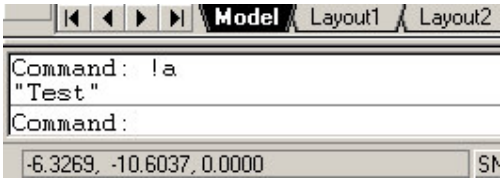
_$ a

De *Console Window* ziet er nu zo uit:



4.1) Command Window versus Console Window

Om de waarde van een variabele op de AutoCAD prompt te bekijken moet je de variabele laten voorafgaan door een uitroepteken (!). In *Visual LISP* type je simpel weg de variabele naam.



Als je in de AutoCAD *Command Window* op de spatie balk drukt wordt de expressie geëvalueerd dan wel het commando uitgevoerd.
In de Visual LISP *Console Window* daarentegen moet je op de *ENTER* toets drukken om de ingevoerde tekst te laten verwerken.
Dit geeft je de volgende mogelijkheden:

- Vervolg de AutoLISP expressie op de volgende regel. Om de expressie op de volgende lijn te vervolgen druk je tegelijk op de *CTRL* en *ENTER* toets.
- Voer meer dan een expressie in voor je op de *ENTER* toets drukt. Visual LISP evalueert eerst alle expressies voordat het een waarde terug naar de Console Window stuurt.

- Als je tekst selecteert in de *Console Window* (bijvoorbeeld het resultaat van een vorige expressie of de expressie zelf) en drukt dan op *ENTER*, dan kopieert Visual LISP de geselecteerde tekst naar de *Console Prompt*.

Zoals al eerder opgemerkt verschillen de *Command Window* en de *Console Window* in de manier waarop ze de TAB en ENTER toetsen verwerken. In de *Console Window* heeft de spatie geen bijzondere betekenis en heeft dus alleen zijn oorspronkelijke functie als opening tussen karakters. In de *Command Window* heeft het drukken op de spatiebalk hetzelfde effect als drukken op de *ENTER* toets (met als uitzondering wanneer je tussen haakjes () werkt): de getypte tekst wordt ogenblikkelijk verwerkt.

4.2) TAB en SHIFT+TAB

Als je geen zin hebt om herhaaldelijk dezelfde tekst in te typen kun je gebruik maken van een andere functie van de *Console Window*: de *TAB*.
Elke keer als je op de *TAB* toets drukt wordt de voorafgaande expressie op de *Console Prompt* afgedrukt. Klaar om te bewerken of om opnieuw uit te voeren. Door herhaaldelijk op de TAB toets te drukken wandel achteruit door de geschiedenis van de Console Window en door op SHIFT + TAB te drukken wandel je weer vooruit. Als je bij de eerst ingevoerde expressie bent gekomen ga je verder met de laatste en herhaal je dus de cyclus. Bijvoorbeeld: stel je hebt de volgende commando's ingetypt:

(setq Oorsprong (getpoint "\nOorsprong van embleem:"))

(setq Straal (getdist "\Straal van embleem:" Oorsprong))

(setq HalveStraal (/ Straal 2))

(setq OorsprongX (car Oorsprong))

(command "_.CIRCLE" Oorsprong Straal)

Om commando's in de *Console Window* terug te halen:

1) Druk een keer op de TAB toets. Visual LISP haalt het laatst ingevoerde commando terug:

$ (command ".CIRCLE" Oorsprong Straal)

2) Druk nogmaals op de *TAB* toets. Het volgende commando wordt op Console Prompt getoond:

_$ (setq OorsprongX (car Oorsprong))

3) Druk weer op *TAB*. Visual LISP toont de vorige regel:

_$ (setq HalveStraal (/ Straal 2))

4) Druk nu eens op *SHIFT + TAB*. Visual LISP toont het volgende commando:

_$ (setq OorsprongX (car Oorsprong))

5) Druk nogmaals op *SHIFT+TAB*. Nu toont Visual LISP de volgende regel:

$ (command ".CIRCLE" Oorsprong Straal)

En dit is het laatste commando dat je op de *Console Prompt* hebt ingevoerd.

6) Druk weer op *SHIFT+TAB.* Omdat het vorige teruggehaalde commando de laatste was die tijdens deze Visual LISP sessie hebt ingevoerd, Visual LISP begint opnieuw met het terughalen van het eerste commando.

_$ (setq Oorsprong (getpoint "\nOorsprong van embleem:"))

Als je dezelfde expressie meerdere keren invoert haalt Visual LISP deze toch maar één keer terug.

Het is mogelijk een associatieve zoekopdracht uit te voeren in de invoer geschiedenis om een specifieke, eerder ingevoerde, opdracht op te zoeken en weer uit te voeren. Om een associatieve zoekopdracht uit te voeren:

1) Typ de tekst die je wilt zoeken. Bijvoorbeeld typ op de *Console Prompt (command* :

_$ (command

2) Druk op *TAB*. Visual LISPzoekt naar text die je hebt ingevoerd dat begint met *(command*:

$ (command ".CIRCLE" Oorsprong Straal)

Als Visual LISP niet vind wat je zoekt, doet het niets (nou ja, misschien geeft het een piepje). Druk op *SHIFT+TAB* om de richting van de associatieve zoekopdracht om te draaien.

4.3) SHIFT+ESC

Om een commando in de *Console Window* te onderbreken druk je op *SHIFT+ESC*. Als je bijvoorbeeld een ongeldige instructie hebt ingevoerd zoals:

_$ ((setq OoorsprongX (car Oorsprong)
((_>

onderbreek je het commando door het drukken van *SHIFT+ESC* en Visual LISP vertoont een "*input discarded"* melding dat lijkt op:

((_>: ; <input discarded>
_$

Als je op de *Console Prompt* een tekst typt, maar niet op *ENTER* drukt en je drukt in plaats daarvan op *ESC* wordt de regel leeg gemaakt. Druk je op *SHIFT+ESC* dan laat Visual LISP de tekst die je hebt ingevoerd staan maar vertoont een nieuwe prompt zonder de tekst te evalueren.

Als je deel van een commando op de *Console Prompt* typt en de AutoCAD window activeert voordat je op *ENTER* drukt krijg je bij terugkeer naar Visual LISP een nieuwe en lege prompt.
De door jouw getypte tekst is zichtbaar in de regel ervoor zodat je die kan kopiëren en plakken, maar je kunt de tekst niet terughalen met *TAB* want hij is niet opgenomen in de *Console* geschiedenis.

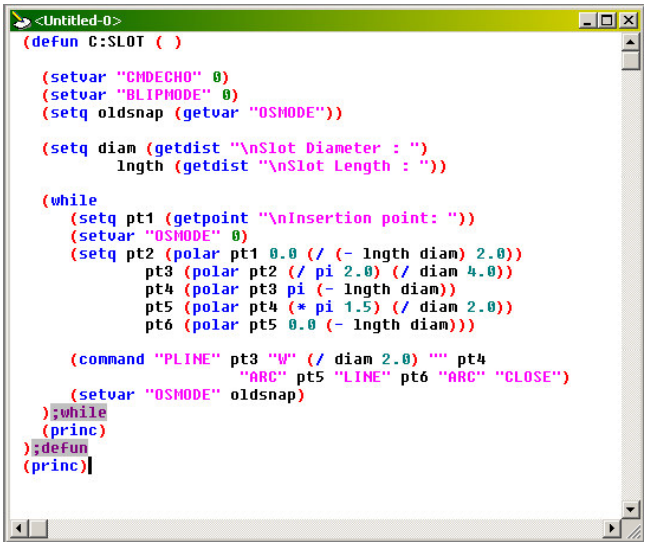
5) Edit Window

5.1) Edit Window openen

Dat is even genoeg gespeeld. Laten we eens wat code laden. Ga naar *FILE→NEW* en kopieer dan deze tekst in de tekst window:

```
(defun C:SLOT ( )

  (setvar "CMDECHO" 0)
  (setvar "BLIPMODE" 0)
  (setq oldsnap (getvar "OSMODE"))

  (setq diam (getdist "\nSlot Diameter : ")
        lngth (getdist "\nSlot Length : "))

  (while
    (setq pt1 (getpoint "\nInsertion point: "))
    (setvar "OSMODE" 0)
    (setq pt2 (polar pt1 0.0 (/ (- lngth diam) 2.0))
          pt3 (polar pt2 (/ pi 2.0) (/ diam 4.0))
          pt4 (polar pt3 pi (- lngth diam))
          pt5 (polar pt4 (* pi 1.5) (/ diam 2.0))
          pt6 (polar pt5 0.0 (- lngth diam)))

    (command "PLINE" pt3 "W" (/ diam 2.0) "" pt4
                  "ARC" pt5 "LINE" pt6 "ARC" "CLOSE")
    (setvar "OSMODE" oldsnap)
  );while
  (princ)
);defun
(princ)
```

De code in de tekst window ziet er nu ongeveer uit als:



5.2) De kleuren van de Edit Window

Voor we verder gaan moeten we even een klein gesprekje over de kleuren van de code in de Edit Window. Zogauw als je tekst plaatst in de *Console Window* of de *Edit Window* probeert Visual LISP te bepalen of het geplaatste woord een ingebouwde functie van AutoLISP is, of een getal, of een een tekst (*string)* of een ander soort taal element. Visual LISP geeft aan elk type element zijn eigen kleur. Dit is zeer handig, want het maakt opzoeken van fouten, zoals missende aanhalingstekens of typefouten in functienamen, en het begrijpen van de code een stuk eenvoudiger.Het standaard kleurenschema staat hieronder:

AutoLisp Taal element	Kleur
Ingebouwde functies en beschermde symbolen	Blauw
Tekst	Magenta
Integere getallen	Groen
Hele getallen	Groenblauw
Commentaar	Magenta op grijze ondergrond
Haakjes openen en sluiten	Rood
Niet herkende elementen (zoals b.v. variabelen)	Zwart

Je kunt uiteraard deze kleuren veranderen, maar doe je zelf een plezier.
Niet doen!

5.3) Tekst selecteren

De eenvoudigste methode om tekst te selekteren is dubbelklikken met je linker muisknop op de tekst. De hoeveelheid tekst die geselecteerd wordt hangt af van de plaats van de cursor.

- Staat de cursor direct voor een open haakje dan wordt alle tekst tot en met het bijpassende gesloten haakje gekozen.

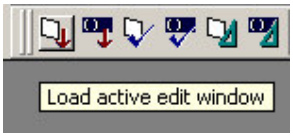
- Staat de cursor direct achter een gesloten haakje dan wordt alle tekst tot en met het bijbehorende open haakje geselecteerd.
- Staat de cursor direct voor of achter een woord, of staat de cursor in het woord dan wordt het alleen het woord gekozen.

Tip: Als je help nodig hebt op een AutoLISP functie dubbelklik dan op de functie om het te selecteren en kies de help button op de menubalk

Help voor deze specifieke functie wordt dan vertoond.

5.4) Programma in geheugen laden

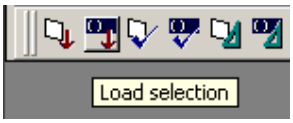
Om het *Slot* programma te laden klik je op de menubutton *Load Active Menu Button*:



Dit laad je code in het geheugen. Om de routine uit te voeren typm je het volgende op de *Console Prompt*:

_$ (c:slot)

Het programma zou nu moeten *runnen,* wanneer nodig schakelend tussen het AutoCAD en de Visual LISP Editor scherm. Als je wilt is het ook mogelijk alleen een gedeelte van de code uit te voeren. Selecteer de code regels die je wilt uitvoeren en klik op de button *Load selection* en druk op *ENTER*.

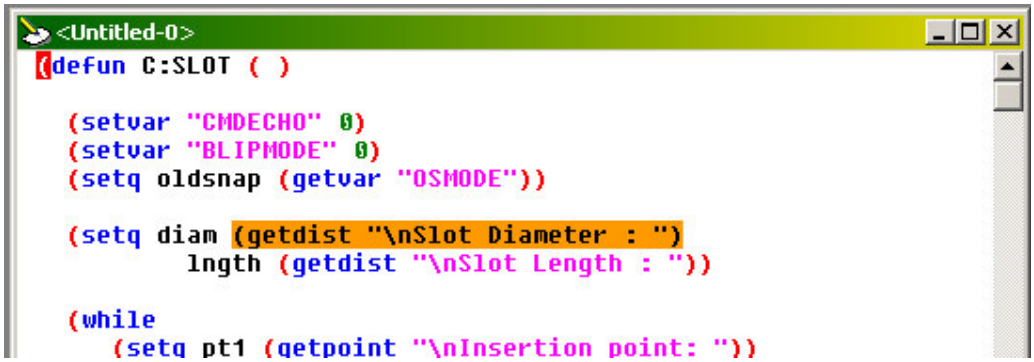


Alleen de code die je geselecteerd hebt wordt uitgevoerd. Een fantastisch hulpmiddel bij het zoeken naar fouten: het zo gehate *debuggen*!

5.5) Foutopsporing c.q. debuggen

Over fout opsporing gesproken, plaats de cursur direct voor de **(defun C:SLOT ()** verklaring. en druk op *F9*. Er wordt nu een *Breakpoint* geplaatst in het programma op de plaats direct achter de cursor. Start het programma opnieuw.

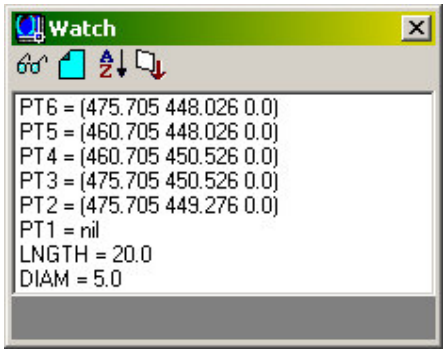
de uitvoering stopt bij het *Breakpoint* teken.Druk nu op *F8*. Als je herhaaldelijk op *F8* drukt loop je stap voor stap door het gehele programma.



Nu een stapje verder: selecteer de *diam* variabele en klik op de *Add Watch* knop van de menubalk:

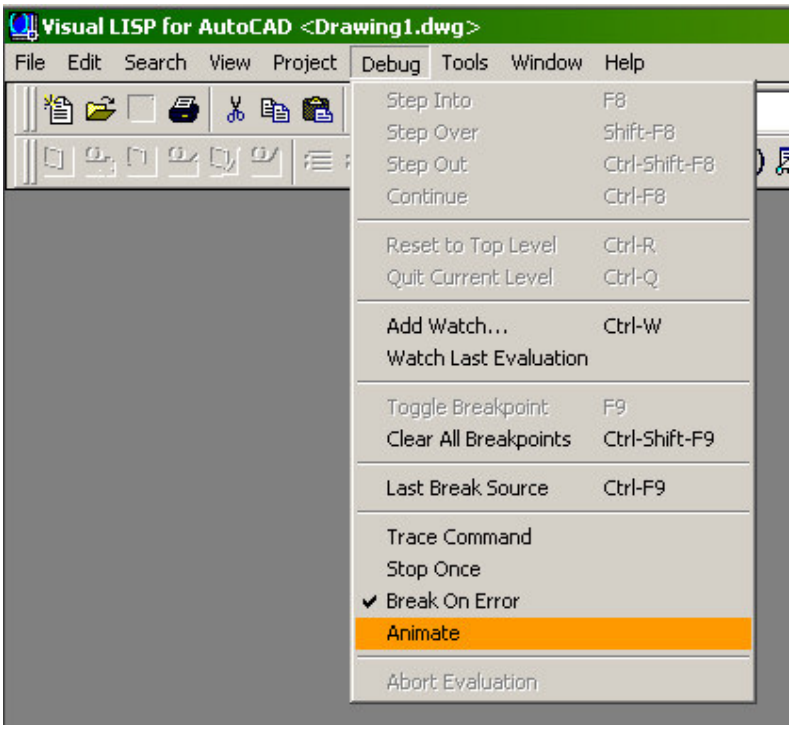
De *Watch dialog box* verschijnt:

Zoals je ziet wordt de variabele *diam* getoond met zijn huidige waarde.
Herhaal het voorgaande voor alle andere variabelen. Het *Watch window* ziet er dan ongeveer zo uit:



Start het programma opnieuw, nog steeds stap voor stap. Merk op hoe de waardes in het *Watch window* veranderen terwijl het programma vordert.

Goed, nu iets leukers. Druk de toeetsen *CRTL+SHIFT+F9* om alle *breakpoints* te verwijderen. Kies nu het menu *Debug* → *Animate*.



Start het programma opnieuw.
Hola! Het programma loopt automatisch. Let op hoe de variabelen in het *Watch window* achtereenvolgens veranderen terwijl het programma doet wat het moet doen.

6) Tot slot

Dit is het dan voorlopig wat de *Visual LISP editor* betreft. Uiteraard heeft de editor veel meer functies dan hier getoond, maar ik heb naar mijn gevoel de meest belangrijke hier besproken. In ieder geval om met programmeren te starten.
In de volgende les gaan we ons eerste programma schrijven.

Tot dan,

Joop Moelee